Opgaveløsninger
(sæt 10)

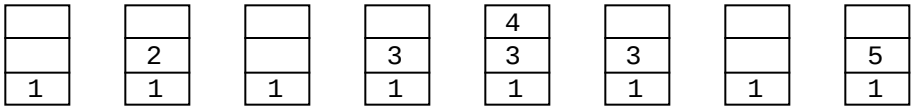
Opgave 34: 15.9 (1 point)

Den første version vil kaste en `ConcurrentModificationException`. Den anden version er korrekt.

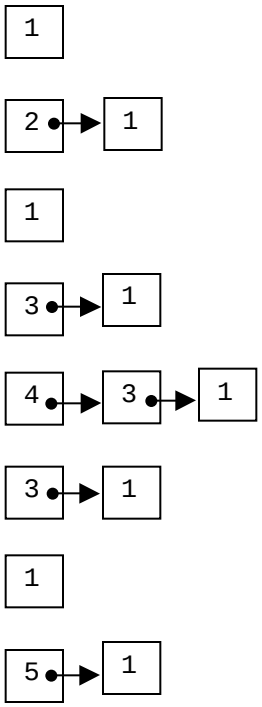
Opgave 35: 16.1 (2 point)

Figurerne nedenfor viser tilstandene for de forskellige datastrukturer.

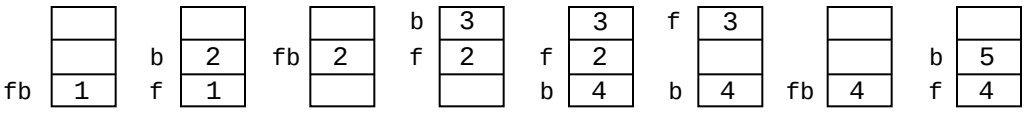
(1) Arraybaseret stak:



(2) Listebaseret stak:

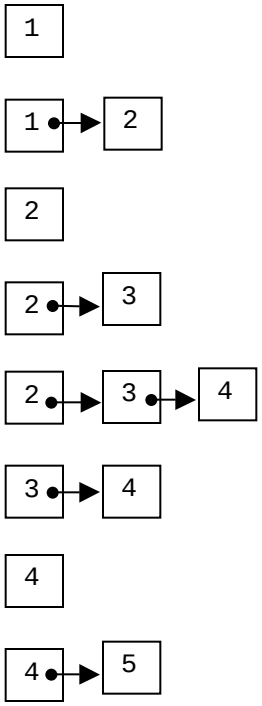


(3) Arraybaseret kø:



`f` betegner `front`
`b` betegner `back`

(4) Listebaseret kø:



Opgave 36: 16.5 (2 point, ikke-obligatorisk)

```
public void addFront(Object x) {
    if (currentSize == theArray.length)
        doubleQueue();
    front = decrement(front);
    theArray[front] = x;
    currentSize++;
}

private int decrement(int x) {
    if (--x == -1)
        x = theArray.length – 1;
    return x;
}

public Object removeBack() throws UnderflowException {
    if (isEmpty())
        throw new UnderflowException("Deque removeBack");
    Object t = theArray[back].element;
    decrement(back);
    currentSize--;
    return t;
}

public Object getBack() throws UnderflowException {
    if (isEmpty())
        throw new UnderflowException("Deque getBack");
    return theArray[back];
}
```

Opgave 37: 17.3 (2 point)

Der benyttes to gennemløb af listen. I første gennemløb vendes `next`-referencerne om. I andet gennemløb løbes listen igennem og dens elementer udskrives. Undervejs i dette gennemløb vendes referencer igen om. Dermed er listen uændret efter udskrivningen.

```
ListNode s = header, t = header.next, u;
while (t != null) {
    u = t.next;
    t.next = s;
    s = t;
    t = u;
}
t = s;
s = null;
while (t != header) {
    System.out.print(t.data);
    u = t.next;
    t.next = s;
    s = t;
    t = u;
}
```

Opgave 38: 17.17 (2 point, ikke-obligatorisk)

```
public LinkedListIterator find(Object x) {
    ListNode n = header;
    while (n.next != null && !n.next.element.equals(x))
        n = n.next;
    if (n.next == null)
        return null;
    if (n != header) {
        Node nodeToMove = n.next;
        n.next = n.next.next;
        nodeToMove.next = header.next;
        header.next = nodeToMove;
    }
    return new LinkedListIterator(header.next);
}
```

Opgave 39: 18.3 (1 point)

```
abbdddbaceeecca
```

Opgave 40: 18.9 (2 point, ikke-obligatorisk)

a)

```
int leaves(BinaryNode t) {
    if (t == null)
        return 0;
    return (t.left == null && t.right == null ? 1 : 0) +
           leaves(t.left) + leaves(t.right);
}
```

b)

```
int oneNonNull(BinaryNode t) {
    if (t == null)
        return 0;
    return ((t.left == null) != (t.right != null) ? 1 : 0) +
           oneNonNull(t.left) + oneNonNull(t.right);
}
```

c)

```
int twoNonNull(BinaryNode t) {
    if (t == null)
        return 0;
    return (t.left != null && t.right != null ? 1 : 0) +
           twoNonNull(t.left) + twoNonNull(t.right);
}
```

De tre algoritmer har alle tidskompleksitet O(N). Hver knude besøges netop én gang.